IN THE SPECIFICATION:

Please amend the following paragraphs in the specification:

Please replace the paragraph on page 9 between lines 6 and 16 with:

FIGURE 6 depicts a code segment 600 illustrating the use of multiple copies of statically stored objects. Specifically, FIGURE 6 illustrates a problem that can occur with static storage if a class is copied into multiple independent compilation units. Classes A and A' for A would be "safe" and would not produce different results than if A were used. However, both A and A' have static variables X, and the getNext operation of both classes increments and returns this value. If only class A is used for all calls to getNext, the values returned will be in monotonically increasing order (6, 7, 8 and so on). However, if class A' is substituted for class A, then for some calls to getNext the order of the returned results will not be as desired. That is, if calls are made sequentially to A, then A, then A', then A', the resulting values will be 6, 7, 6, 7. This problem is addressed by the invention.

Please replace the paragraph on page 10 between lines 3 and 10 with:

FIGURE 8B depicts a reentrant static addressing scheme utilizing the code segment of FIGURE 8A. Specifically, FIGURE 8B illustrates the traditional reentrant scheme for addressing static storage. In this case a register 850 is specifically loaded to address static storage, and all references 840 to static are indirect via that register 850. This allows there to be multiple copies of static, one for each process sharing the common code address space. Even though the code address space is shared, the various processes run without interfering with each other, since each has its own private copy of static storage.

Please replace the two paragraphs starting on page 15 at line 30 and ending on page 16 at line 15 with:

FIGURE 12 depicts a code segment useful in understanding the invention. Specifically, FIGURE 12 depicts an indirect call code segment 1210 and a direct call code segment 1220 as implemented on an AS/400 computer. The value loaded into a

register RS is the address of the basic class description structure 1230. Prepended to the basic class description structure is the constant pool vector table 950. Negative offsets from register RS can be used to access entries in this table 950. Entries in this table 950 contain pointers to the actual constant pool entries stored in heap.

In the indirect call case 1210, the constant pool entry for the target method is referenced by using a negative offset from RS (the hardware of the AS/400 supports negative offsets in its instructions) to fetch the pointer to the MethodRef constant pool entry (into register RA in this example). The index of the pointer within the vector table 950 is fixed at compile time, or in the case of constant pool entries that derive directly from the class file, is fixed when the class file is created. Thus a "hard coded" negative offset can be used. Next, the address of the MethodInfo structure (i.e., the address of the method table entry) for the target method is loaded into RB, using base register RA and the known offset from the start of a MethodRef entry to the MethodInfo pointer within that entry. Then, the entry point address of the method is loaded into register RC, using base register RB and the known offset from the start of a MethodInfo entry to the entry point value.

Please replace the four paragraphs starting on page 17 at line 15 and ending on page 18 at line 8 with:

FIGURE 14 depicts a graphical representation useful in understanding the invention. Specifically, FIGURE 14 depicts the block diagram of some of the data structures involved in call sequences. A method table entry 1410 (or MethodInfo) contains method identification 1411, a pointer to the basic class description 1412 of the containing class 1430, a pointer to the method's entry point 1413, a pointer to the method's code gen static 1414 (i.e., static storage generated by the low-level code generation process), debug, info, information descriptive of the method 1416 (authorities, etc,) and other miscellaneous information 1417.

An object 1420 contains a pointer of the basic class description 1421 of the class 1430 of which the object is an instance, some object status information 1422 (e.g., lock state), and the actual object data 1423 corresponding to the object fields declared in the Java, source.

The basic class description 910–, as described previously, contains several descriptive pieces of information for the class 1430, including pointers to other areas containing the remaining full class description. Prepended to the front of the basic class description is the constant pool vector table 950, consisting of pointers to various types of constant pool entries. The most important of these entries are described below with respect to FIGURE 15. One of the constant pool entry types is the MethodRef 1510, an entry which corresponds to an externally referenced method and which contains, indirectly, the information required to reference the method.

Postpended to the basic class description 910 is the virtual method table 1070 of the target method's class, not in the calling method's class. This is a table of pointers to method table entries 1440 (MethodInfo entries 1440-1, 1440-2, and 1440-3) that reference the methods of the class 1430. The entries 1440 are ordered with those inherited from superclasses first, and in the same order as they appear in the table of the superclass. This allows efficient virtual method calls to be made without first having to look up the method name in the particular class whose object is being used to base the call.

Please replace the paragraph on page 19 between lines 13 and 14 with:

The UTF8 entry 1540 includes an identification tag ~~1542~~1541, various flags 1542, a pointer to the appropriate data 1543 and an indication of the length of the data 1544.

Please replace the paragraph on page 20 between lines 27 and 28 with:

At step 1632, version information associated with the class is checked. At step 1634, a determination is made as to whether a version information mismatch exists. The error is handled 1636 if the mismatch exists. Otherwise, the process is done 1638.

Please replace the paragraph on page 21 between lines 23 and 29 with:

The clone basic class description 1714 includes a first linkage 1714A to the clone full class description 1730, and a second linkage 1714B to the parent basic class description 1724. The parent basic class description 1724 includes an entry

~~724A~~1724A having a linkage to the parent full class description 1740. The field references 1713 and 1723 of, respectively, the clone basic class description 1710 and the parent basic class description 1720 have linkages to the Java® static storage 1743. The method reference 1722 has a linkage to the parent method table 1742.